

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

EV205822539

**Variable Play Speed Control for Media Streams**

**Inventors:**

Li-wei He

Adil Sherwani

Patrick N. Nelson

Richard William Saunders

Jonathan M. Cain

Anders E. Klemets

ATTORNEY'S DOCKET NO. MS1-1544US

## **TECHNICAL FIELD**

The present disclosure generally relates to streaming media, and more particularly, to providing variable play speed control for playing back media streams.

## **BACKGROUND**

As the popularity of playing multimedia content over the Web has increased the methods for accessing such multimedia content have continually improved. Initially, playing multimedia content (e.g., audio and video) was primarily a download-and-play technology. This method requires that an entire media file be downloaded from a Web server before it can be played. Thus, a media file becomes a local file on a client computer prior to being played back on the client. Because media files are typically quite large, however, the download-and-play method can require a significant amount of time to download a media file before the file can be played back.

Another method for accessing multimedia content, called a progressive download, also uses a standard Web server to supply data (e.g., a compressed media file) to a client. In this method, however, the client begins playing back the media file before the entire file is fully downloaded from the server. Thus, the time between a media selection and the beginning of playback is typically much shorter with this method than with the download-and-play method previously discussed. Playback of the media file begins during the streaming of the file once the client has buffered a few seconds of content. The buffering provides a small backlog of information so the media can continue to play uninterrupted even during periods of high network congestion. With the progressive download

1 delivery method, the client retrieves data as fast as the Web server, the network and  
2 the client will allow without regard to the bit-rate parameter of the compressed  
3 media stream.

4 Streaming media servers provide still another method for accessing  
5 multimedia content. In the streaming media server method, a compressed media  
6 file is stored on a specialized streaming media server instead of a Web server.  
7 Unlike a Web server, which simply delivers data as fast as it can, a streaming  
8 media server can actively and intelligently send data to a client. The data is  
9 delivered at the data rate associated with the compressed media streams (e.g.,  
10 audio and video streams), which is the exact real-time rate at which the data will  
11 be played back. The server and client communicate during the delivery process  
12 and the server can respond to feedback from the client. Among other benefits, the  
13 streaming media server's "just-in-time" manner of delivering data preserves  
14 network bandwidth that can be used to service more clients.

15 One important aspect of accessing media content, regardless of the method  
16 of delivery, is the ability to navigate the content and/or find specific locations  
17 within the content. However, the current methods discussed above for  
18 accessing/delivering multimedia content have significant disadvantages in this  
19 regard. For example, although some media players provide navigation functions  
20 such as fast forward and rewind, content delivery systems (e.g., Web servers,  
21 streaming media servers) may not support such accelerated or decelerated  
22 playback. Web servers, for example, are not configured to comprehend a client  
23 request for accelerated playback. In addition, even when streaming media servers  
24 support accelerated playback (or decelerated playback), the ability of a user to  
25 comprehend the content at the accelerated rate is greatly diminished because

1 traditional streaming media servers simply drop data from media streams and only  
2 send "key frame" of video to achieve the accelerated rate. Thus, there is no true  
3 acceleration through the content, rather, there is a "skipping" through the content.  
4 For example, a fast forward request (e.g., a request for 5 times the normal/real-  
5 time delivery/playback rate) from a client might result in the streaming media  
6 server sending only 1 video frame for every 8 seconds worth of content. This is  
7 approximately equivalent to dropping 239 out of every 240 video frames from a  
8 video stream. Thus, fast forwarding results in a jerky effect as if a sequence of still  
9 images is being delivered. In addition, traditional streaming media servers  
10 typically drop the entire audio stream from the media content if asked to accelerate  
11 content delivery, because the servers assume there is not enough bandwidth to send  
12 the entire stream over the network at 5 times the real-time playback rate. Also,  
13 client based media players typically drop the audio stream when fast forwarding,  
14 even when playing a local file, because they assume that the fast forwarded audio  
15 playback produces high-pitched, "chipmunk" sounding audio that is mostly  
16 incomprehensible. Furthermore, any non-continuous, non-video/audio data stream  
17 (e.g., script commands for triggering events, captions, metadata) included within  
18 the media content and synchronized to play at particular times during video  
19 playback is typically lost due to the skipping through of the video content.

20 One attempt to address the problems with navigating media content has  
21 been the development of "add-ons" for client media players. Add-ons are software  
22 additions that can be added onto an existing media player to provide an improved  
23 media content navigation experience. Although such add-ons may provide some  
24 benefits under certain circumstances, they have significant disadvantages. For  
25 example, such add-ons can provide an accelerated playback only when the media

1 content is present in a local media file residing on the client computer. Thus, the  
2 drawbacks of the download-and-play method discussed above apply. Add-ons  
3 generally operate by tricking the underlying media player engine into consuming  
4 data at a faster rate while providing no mechanism for requesting accelerated  
5 delivery from a content delivery system (e.g., a streaming media server, a Web  
6 server). Thus, if the media content is not already presently available at the client in  
7 a local file, playback can only occur as fast as data arrives from a streaming media  
8 server or Web server. Therefore, use of add-ons when the media source is a  
9 streaming media server results in playback at the data rate associated with the  
10 compressed media stream being delivered to the client computer. When a standard  
11 Web server is the media source, use of an add-on can result in playback at rates  
12 that are various and unknown because the data delivery rate from the Web server  
13 depends on momentary network bandwidth availability and other varying factors.  
14 This can make it difficult or impossible to comprehend the media content. In  
15 addition, such add-ons provide no control over other functions of a media player  
16 because they are not an integral part of the player. Thus, use of an add-on can  
17 result in a loss of other basic controls on a media player such as "play", "stop" and  
18 "pause".

19 Accordingly, a need exists for an integrated and comprehensive solution  
20 capable of supporting variable play speed control for media streams.

## 21 22 **SUMMARY**

23 Variable play speed control of media streams is described herein.

24 In accordance with one implementation, a media stream is received from a  
25 source. The source of the media stream is determined. Whether or not the source

1 can deliver the media stream at an accelerated rate is also determined. Variable  
2 play speed controls are enabled or disabled depending on the source and on  
3 whether the source can deliver the media stream at the accelerated rate.

4 In accordance with another implementation, media content is requested  
5 from a source at an accelerated rate. The accelerated rate is a rate that exceeds a  
6 normal playback rate for the media content. A media stream is received that  
7 includes an uninterrupted data stream of the media content from which no data has  
8 been intentionally dropped. All of the content from the media stream is rendered  
9 at the accelerated rate.

10 In accordance with another implementation, a media player includes  
11 variable play speed controls to vary playback speed of a media stream. The media  
12 player additionally includes a playback module to enable or disable the variable  
13 play speed controls depending on the source of the media stream and whether the  
14 source can deliver the media stream at a requested rate, a graphical user interface  
15 (GUI) module to support a GUI for presenting the variable play speed controls to a  
16 user, and an application programming interface (API) to expose the variable play  
17 speed controls to the programmatic control of a custom application program.

## 18 19 **BRIEF DESCRIPTION OF THE DRAWINGS**

20 The same reference numerals are used throughout the drawings to reference  
21 like components and features.

22 Fig. 1 illustrates an exemplary network environment suitable for  
23 implementing variable play speed control of media streams.

1 Fig. 2 illustrates an exemplary embodiment of a client computer suitable for  
2 implementing variable play speed control of media streams in conjunction with  
3 various media content sources.

4 Fig. 3 illustrates an example of a playback filter graph.

5 Fig. 4 illustrates an example of a graphical user interface showing various  
6 playback controls of a media player.

7 Fig. 5 illustrates an example of a filter graph that has been assembled with a  
8 pitch adjustment filter.

9 Fig. 6 illustrates an example of media content having a video data stream,  
10 an audio data stream and a non-audio/video data stream.

11 Figs. 7 - 9 illustrate block diagrams of exemplary methods for implementing  
12 variable play speed control of media streams.

13 Fig. 10 illustrates an exemplary computing environment suitable for  
14 implementing a client computing device and a content server computing device.

## 15 16 **DETAILED DESCRIPTION**

### 17 **Overview**

18 The following discussion is directed to systems and methods that support  
19 variable play speed control for media streams. The variable play speed control for  
20 media streams discussed herein provides an end-to-end solution for media stream  
21 delivery, playback, and user interface that enables end users and software  
22 developers to dynamically control the playback speed of media streams without  
23 losing the ability to comprehend the media content.

24 The variable play speed control for media streams combines multiple  
25 features into a single integrated end-to-end solution that provides advantages

1 including fully rendered media content at accelerated and decelerated rates.  
2 Rendering all the media content (i.e., not skipping video content or leaving out  
3 audio content) improves a user's ability to comprehend the content at accelerated  
4 or decelerated rates. In addition, rendering all the media content permits true time  
5 compression at accelerated rates which reduces the amount of time it takes to  
6 consume the content. Furthermore, rendering all the content allows for fully  
7 rendering all non-audio/video data within media streams such as script commands  
8 for triggering events and other data streams such as captions and metadata.

9 Other advantages of the disclosed variable play speed control for media  
10 streams include audio pitch adjustment to improve a user's ability to comprehend  
11 accelerated and decelerated audio content, a graceful degradation of playback  
12 quality (e.g., rendering only video or video key frames) in circumstances where a  
13 connection or bandwidth do not allow all the content to be rendered, a built-in  
14 streaming media platform enabling third party developers to access and take  
15 advantage of the variable play speed control, and the ability to implement variable  
16 play speed control on media streams from a variety of sources including streaming  
17 media servers.

### 18 **Exemplary Environment**

19  
20 Fig. 1 illustrates an exemplary network environment 100 suitable for  
21 implementing variable play speed control of media streams. In the exemplary  
22 network environment 100, multiple client computing devices 102 are coupled to  
23 multiple streaming media servers 104 and/or multiple standard Web servers 106  
24 via a network 108. Network 108 is intended to represent any of a variety of  
25 conventional network topologies and types (including optical, wired and/or

1 wireless networks), employing any of a variety of conventional network protocols  
2 (including public and/or proprietary protocols). Network 108 may include, for  
3 example, the Internet as well as possibly at least portions of one or more local area  
4 networks (LANs) and/or wide area networks (WANs).

5       Requests from a client computer 102 for streaming media content are routed  
6 from the client computer 102 to a streaming media server 104 or a standard Web  
7 server 106 via network 108. In general, servers 104 and 106 receive requests and  
8 return the requested content to the requesting client computer 102 via network 108.  
9 More specifically, a media file's URL (Uniform Resource Locator), typically  
10 located on a Web page, can be activated to launch a client-side 102 media player  
11 and download (i.e., from a Web server 106) or stream (i.e., from a streaming media  
12 server 104) the media file to the client 102.

13       The data in a media file is typically delivered as a compressed media data  
14 stream and can include any of a variety of one or more types of content, such as  
15 audio, video, text, images, animation, and so on. The data may be publicly  
16 available or alternatively restricted (e.g., restricted to only certain users, available  
17 only if the appropriate fee is paid, etc.). Additionally, the data may be "on-  
18 demand" (e.g., pre-recorded and of a known size) or alternatively "broadcast" (e.g.,  
19 having no known size, such as a digital representation of a concert that is captured  
20 live as the concert is performed and made available for streaming shortly after  
21 capture).

22       Delivery (i.e., streaming) of media content from a Web server 106 uses the  
23 Hyper Text Transport Protocol (HTTP) which is the standard Web protocol used by  
24 Web servers and Web browsers for communication between the server 106 and the  
25 client 102. HTTP operates on top of the Transmission Control Protocol (TCP),

1 which handles all the data transfers. TCP is optimized for non-real-time  
2 applications such as file transfer and remote log-in. An objective of TCP is to  
3 maximize the data transfer rate while ensuring overall stability and high  
4 throughput of the entire network 108. When sending data from a server 106 to a  
5 client 102, TCP first sends data at a low data rate and then gradually increases the  
6 rate until the client 102 reports a data packet loss. TCP then assumes it has hit the  
7 bandwidth limit or network congestion, and starts over by sending data at a low  
8 data rate. It then gradually increases the data rate and repeats the process. Thus,  
9 delivery of media content from a Web server 106 to a client 102 means that the  
10 Web server 106 delivers (and the client computer 102 receives) data as fast as the  
11 Web server 106, the network 108 and the client computer 102 will allow without  
12 regard to the bit-rate parameter of the compressed media data stream.

13 By contrast, a streaming media server 104 actively and intelligently  
14 manages data delivery to the client computer 102. Thus, the streaming server 104  
15 can deliver media content at the exact data rate associated with the compressed  
16 media data streams (e.g., the compressed audio and video streams). The server  
17 104 and client 102 stay in close touch during the delivery process, and the  
18 streaming media server 104 can respond to requests from the client. Therefore, the  
19 server 104 can also deliver media content at varying data rates requested by the  
20 client 102 as discussed in greater detail herein below. While streaming media  
21 servers 104 can use the HTTP/TCP protocols used by Web servers 106, they can  
22 also use specialized protocols such as the User Datagram Protocol (UDP) to  
23 improve the streaming experience. UDP is an ideal protocol for transmitting real-  
24 time audio and video data.

1 In addition to a Web server 106 and a streaming media server 104 as  
2 sources of media content, a local storage medium on the client computer 102 itself  
3 can be a streaming media source. Media content can be delivered from a local  
4 storage medium on the client computer 102 to a media player on the client  
5 computer 102. In this case, media content would be sourced from a local media  
6 file that would typically have been previously downloaded from a Web server 106  
7 or otherwise stored on the client computer 102.

8 The client computer 102, streaming media server 104, and Web server 106  
9 can each be any of a variety of conventional computing devices, including desktop  
10 PCs, notebook or portable computers, workstations, mainframe computers, Internet  
11 appliances, gaming consoles, handheld PCs, cellular telephones or other wireless  
12 communications devices, personal digital assistants (PDAs), combinations thereof,  
13 and so on. One or more of devices 102, 104 and 106 can be the same types of  
14 devices, or alternatively different types of devices. An exemplary computing  
15 environment for implementing a client computer 102, a streaming media server  
16 104, and a Web server 106 is described in more detail herein below with reference  
17 to Fig. 10.

18 The implementation of variable play speed control for media streams with  
19 respect to each of the media sources mentioned above (i.e., a Web server, a  
20 streaming media server, and a local medium) is discussed in greater detail below  
21 with regard to exemplary embodiments.

### 22 23 **Exemplary Embodiments**

24 Fig. 2 illustrates an exemplary embodiment of a client computer 102  
25 suitable for implementing variable play speed control of media streams received

1 from various sources. The media streams may be delivered to client computer 102  
2 from various media file sources that include a Web server 106, a streaming media  
3 server 104, and a local media file 200 previously stored on a storage medium (e.g.,  
4 a hard disc, not illustrated in Fig. 2) of client computer 102. In general, media  
5 streams from source media files (e.g., 200, 202, 204) can be played by a media  
6 player 206 configured to execute on client computer 102.

7 Source media files such as files 200, 202 and 204 can be streamed (i.e.,  
8 delivered) to a client computer 102 in accordance with any of a variety of different  
9 streaming media formats. These formats can include audio formats, audio-video  
10 formats, or various other formats now existing or yet to be created by a content  
11 provider. For example, media can be streamed in accordance with the ASF format  
12 (Advanced Systems Format or Advanced Streaming Format). Additional  
13 information regarding ASF is available from Microsoft® Corporation of Redmond,  
14 Washington. Alternatively, or in conjunction with the ASF format, other streaming  
15 media formats may be used such as WMA (Windows Media Audio), WMV  
16 (Windows Media Video), MPEG (Moving Pictures Experts Group)-1, MPEG-2,  
17 MPEG-4, Quicktime, and so on.

18 Accordingly, client computer 102 of Fig. 2 includes a media player 206 and  
19 at any given time may also include a local media file 200, a progressively  
20 downloaded media file 208, and a media stream cache 210 of a streamed media  
21 file. Traditionally, there are three ways of delivering “streaming media” (e.g.,  
22 media files having audio and video data) to an end user operating a media player  
23 206 on a client computer 102. The first is through a local media file 200 that has  
24 been previously loaded onto a storage medium of client computer 102, such as a  
25 hard disc. There are various ways a local media file 200 can be loaded onto client

1 computer 102 including, for example, from a portable storage medium (e.g., floppy  
2 disc, optical disc, memory stick, etc.) inserted into client computer 102 or through  
3 a download from a standard Web server 106. Once the local media file 200 is  
4 completely loaded onto client computer 102, it can be played by a media player  
5 206 directly from the storage medium of the client computer 102.

6 A second way to deliver streaming media to a client computer 102 is  
7 through a progressive download of a media file 202 from a standard Web server  
8 106. Media files 202 are typically stored on, and downloaded from, a Web server  
9 106 in a compressed format. The media file 202 is saved locally as a progressive  
10 download media file 208 in a manner similar to the download-and-play method of  
11 the local media file 200 discussed above. However, in the progressive download  
12 method, while the streaming media file 202 is being delivered from the Web server  
13 106, the media player 206 on client computer 102 begins playing the media content  
14 (e.g., audio and/or video streams) after a few seconds of buffering. Thus, the  
15 client 102 implements a “progressive playback” as the Web server 106  
16 “progressively downloads” the media file. The buffering provides a small backlog  
17 of data that allows the media to continue playing uninterrupted even during periods  
18 of high network 108 congestion. When media is streamed from a standard Web  
19 server 106 as a progressive download, the Web server 106 delivers the data (and  
20 the client 102 receives the data) as fast as the Web server 106, the network 108 and  
21 the client computer 102 will allow, without regard to the bit-rate parameter of the  
22 compressed media data stream.

23 A third way to deliver streaming media to a client computer 102 is from a  
24 media file 204 on a streaming media server 104. Like media files 202 on a  
25 standard Web server 106, media files 204 on a streaming media server 104 are

1 typically stored and streamed in a compressed format. As mentioned above,  
2 streaming media servers 104 actively and intelligently manage delivery of media  
3 data to a client computer 102. Although streaming media servers 104 typically  
4 deliver media content at the exact data rate associated with a compressed media  
5 file 204, the streaming media server 104 in the embodiment of Fig. 2 is  
6 additionally capable of delivering media content at varying rates according to  
7 requests made from client computer 102. In general, a variable speed streaming  
8 module 212 on streaming media server 104 communicates with media player 206  
9 on client computer 102 and responds to requests to deliver media files 204 at  
10 various data rates.

11 Traditionally, when a media file is streamed from a streaming media server  
12 to a client computer, the media file is played directly from the network 108 as it  
13 arrives at the client computer. Thus, the streaming media data is not saved locally  
14 on the client computer. However, in an embodiment of Fig. 2, a media stream  
15 cache 210 is included on client computer 102. The media stream cache 210  
16 supports variable play speed scenarios of the current embodiment in which a live  
17 broadcast can be cached so that playback can be accelerated from the beginning of  
18 the content in order to "catch up" with the live broadcast. Variable play speed  
19 control of the media player 206 is discussed in greater detail below.

20 Media player 206 of client computer 102 includes various modular software  
21 components including variable play speed controls 214, basic transport controls  
22 216, playback filter graph 218, playback module 220, graphical user interface  
23 (GUI) module 222 and media player application programming interface (API) 224.  
24 It is noted that these components are illustrated as part of media player 206 for  
25 purposes of illustration and discussion and not for purposes of limitation. In

1 general, such components comprise various modules (or combinations of modules)  
2 having computer/processor executable instructions that may be located in one or  
3 more memories (not illustrated in Fig. 2, but see Fig. 10) of client computer 102.

4 The media player 206 generally controls and processes streaming media  
5 data from a source media file (e.g., media files 200, 202, 204) through one or more  
6 playback graphs 218. As illustrated in Fig. 3, for example, a playback graph 218  
7 includes modular functional components called filters that are graphed together to  
8 process media data in particular ways according to particular media data types and  
9 user playback preferences. A playback graph typically includes filters that can be  
10 categorized into one of three filter types: a source filter, a transform filter, and a  
11 rendering filter. A source filter 300 accepts and reads data from a source media  
12 file, such as a source files 200, 202 and 204 that may be delivered from a local  
13 storage medium, a Web server 106, or a streaming media server 104. Thus, the  
14 source filter 300 introduces the source data into the playback graph 218.

15 A transform filter, such as splitter filter 302 and audio and video  
16 decompression filters 304 and 308, accepts data from the source filter 300,  
17 processes the data, and forwards the processed data to a rendering filter (e.g.,  
18 filters 306, 308 and 310). Transform filters can encompass a variety of filter types  
19 such as a splitter filter 302 which splits a single media data stream into component  
20 audio, video, and other data streams. Audio decompression filter 304 and video  
21 decompression filter 308 are transform filters that decompress data streams  
22 delivered from compressed media files such as files 200, 202 and 204. Various  
23 types of transform filters can be alternately included in a playback graph 218 to  
24 cause a particular desired effect in the playback of the rendered data streams. One  
25 such filter is an audio pitch adjustment filter that is discussed in greater detail

1 below with reference to variable play speed controls and the playback graph 218 of  
2 Fig. 5.

3 A rendering filter (e.g., audio rendering filter 306, video rendering filter  
4 310, synchronized data rendering filter 312) renders data to a form that is useful in  
5 driving a hardware device such as an audio speaker 314 or a video display screen  
6 316. Thus, rendered output is typically supplied to a hardware device (e.g.,  
7 speaker 314, display screen 316), but could also be supplied to any location that  
8 accepts media input (such as a file maintained on a volatile memory, optical disk,  
9 hard disk, etc.). In general, media players typically include audio and video  
10 rendering filters that comprehend audio and video data types. However, various  
11 other types of rendering filters from 3<sup>rd</sup> party software developers, for example,  
12 might also be loaded into a media player graph 218 to enable the player to render  
13 previously unknown and custom data types.

14 It is noted that the playback graph 218 shown in Fig. 3 represents only one  
15 of many possible constructions of playback graphs and is not intended as a  
16 limitation on the architecture of playback graphs in general. Furthermore,  
17 although three basic types of filters are described above, those skilled in the art  
18 will appreciate that a filter can represent a combination of different filter types.

19 Accordingly, playback graphs 218 can vary in their complexity and  
20 configuration for any given set of media data types and playback instructions  
21 entered by a user through, for example, media player controls 214 and 216. The  
22 playback module 220 of media player 206 performs various functions related to the  
23 playback of media data including controlling the assembly of the playback graph  
24 218 and managing the flow of data streams within the playback graph 218 by  
25

1 directing the movement of data through the filter components of the playback  
2 graph 218.

3 The playback module 220 supports the construction of a playback graph  
4 218 by locating enabled filters capable of appropriately processing a particular  
5 media type in a particular manner. Thus, among other things, the playback module  
6 220 determines a media type for a data stream received by the media player 206  
7 and determines appropriate filters that are available for processing the data stream.  
8 The playback module 220 constructs a playback filter graph 218 by connecting  
9 filter components into a series of filters beginning with a source filter and ending  
10 with a rendering filter as discussed above with reference to the playback graph 218  
11 of Fig. 3. Additional functions of the playback module 220 related to variable play  
12 speed controls are discussed herein below.

13 As illustrated in Fig. 2, the media player 206 supports various playback  
14 controls 214 and 216 for controlling media playback. The controls include  
15 variable play speed controls 214 and basic transport controls 216. The variable  
16 play speed controls include a play speed control, a content seek control, a fast  
17 forward control, a rewind control, and a next frame and prior frame control. The  
18 basic transport controls 216 include basic controls for playing, pausing, and  
19 stopping media playback.

20 The underlying playback controls (214 and 216) are presented to an end  
21 user through a graphical user interface (GUI) 226 that is supported by a GUI  
22 module 222. The GUI 226 is displayed on a display device 228. Display device  
23 228 is typically implemented as a display monitor that is a peripheral device  
24 coupled to a client computer 102. However, for purposes of discussion, display  
25 device 228 has been illustrated in Fig. 2 as being a part of client computer 102. A

1 user has access to playback controls (214, 216) through the GUI 226 presented on  
2 display device 228 and through an input device such as a mouse or keyboard (not  
3 illustrated in Fig. 2).

4 Fig. 4 shows an example of a GUI 226 that generally illustrates the various  
5 playback controls (214 and 216) of media player 206. The play speed control 400  
6 allows a user to dynamically control the playback rate of a media stream being  
7 played by media player 206. A user can “grab” the play speed control 400 (e.g., by  
8 clicking and holding with a mouse) and drag it to different play speed settings  
9 along a graduated playback rate bar 402. By doing so, a user can speed up  
10 playback of a media stream beyond a normal, real-time playback rate and slow  
11 down playback below the normal, real-time playback rate. A play speed setting of  
12 1.0 indicates the normal, or real-time, playback rate that is intended for consuming  
13 the media. The graduated playback rate bar 402 shows a range of “-16 times” the  
14 real-time rate to “16 times” the real-time rate. However, the graduated playback  
15 rate bar 402 and corresponding rate numbers are also highlighted in a manner that  
16 indicates to a user that the range of playback rates in which the media can best be  
17 comprehended is between 0.5 times the real-time rate and 2.0 times the real-time  
18 rate. It is noted that the rate numbers presented along the graduated playback rate  
19 bar 402 are intended as examples only, and while they may be a realistic  
20 illustration of useful rate numbers, they are not intended to be a limitation as to the  
21 range of rates that might be controlled by the play speed control 400 of media  
22 player 206.

23 The content seek control 404 can also be “grabbed” (e.g., by clicking and  
24 dragging with a mouse) and moved to different locations along a content location  
25 bar 406. Moving the content seek control 404 moves a user to different locations

1 in a media selection relative to the position of the seek control 404 along the  
2 content location bar 406. The next and prior frame controls 408 step a user frame  
3 by frame, either forward or backward, through a video presentation. The fast  
4 forward 410 and rewind 412 controls speed a user through a media selection, either  
5 forward or backward, in a manner similar to that for the play speed control 400.  
6 Also shown on the GUI 226 of Fig. 4 are the basic controls 216, play/pause 414  
7 and stop 416.

8 Each of the variable play speed controls 214 just discussed is configured to  
9 initiate some measure of acceleration or deceleration of the playback rate of media  
10 streams being processed through a playback graph 218 on media player 206.  
11 Moreover, in addition to accelerating or decelerating the playback rate of the  
12 media streams in the playback graph 218, each of the variable play speed controls  
13 214 is configured to initiate a request through the playback module 220 to  
14 accelerate or decelerate the delivery rate of media streams to the media player 206.  
15 Therefore, in addition to controlling the playback graph 218, the playback module  
16 220 communicates with media file sources in order to request varying delivery  
17 rates for streaming media according to user input via variable play speed controls  
18 214.

19 In order to communicate with a media file source, the playback module 220  
20 first determines the source of the media file. The playback module 220 is  
21 configured to query the source layer 229 for information about the source type, for  
22 example, to determine if the media source is a local media file 200, a progressive  
23 download media file 208 from a Web server 106, or a media stream 210 from a  
24 streaming media server 104. Queries from the playback module 220 regarding  
25 data delivery rates from sources that are not local (i.e., progressive download

1 media files 208 from a Web server 106, or media streams 210 from a streaming  
2 media server 104) are delegated to the network layer 230. The network layer 230  
3 is not invoked at all for local media content.

4 The playback module 220 is also configured to enable or disable the  
5 variable play speed controls 214 of the media player 206 based on particular  
6 circumstances that indicate whether or not delivery of data at a variable rate is  
7 possible. For example, delivery at variable rates is not possible if the media source  
8 is a standard Web server 106 or if prohibitive network bandwidth limitations exist.  
9 Thus, the playback module 220 determines the source of the media file and  
10 determines if the source is capable of delivering data at a variable rate. Based on  
11 these determinations, the playback module 220 disables or enables the variable  
12 play speed controls 214. Furthermore, the GUI module 222 supports these  
13 changes in operability of the variable play speed controls 214 by altering the  
14 appearance of the controls 214 on the GUI 226 as they are presented on display  
15 device 228. The changes typically manifest themselves through the GUI 226 as  
16 coloration differences in the controls 214 that indicate when the controls 214 are  
17 enabled or disabled. Therefore, a user is aware of when the variable play speed  
18 controls 214 are operable and when they are inoperable.

19 In the case where the media source is a local media file 200, the variable  
20 play speed controls 214 remain enabled because there is not presumed to be a limit  
21 on the speed at which data from the local media file 200 can be delivered to the  
22 media player 206. Therefore, the playback module 220 services requests for  
23 variable play rates initiated by a user from the variable play speed controls 214 by  
24 controlling the playback graph 218 to accelerate or decelerate the media data.  
25

1 Thus, the playback module 220 maintains the variable play speed controls 214 in  
2 an enabled status and the user is able to manipulate the controls from the GUI 226.

3 In the case where the playback module 220 determines that the media  
4 source is a "progressive download" from a Web server 106, it initially disables the  
5 variable play speed controls 214. As mentioned above, Web servers 106 are  
6 configured to "progressively download" data as fast as the Web server 106, the  
7 network 108 and the client computer 102 will allow, without regard to the bit-rate  
8 parameter of a compressed media data stream. Web servers 106 are not configured  
9 to comprehend requests regarding variable data delivery rates. Therefore, when  
10 the playback module 220 queries the source layer 229 and determines that the  
11 media source is a Web server 106, the playback module 220 disables the variable  
12 play speed controls 214 on the media player 206 until such time as the entire media  
13 file 202 has been downloaded as a progressive download media file 208 onto the  
14 client computer 102. Thus, the variable play speed controls 214 will be inoperable  
15 during the progressive download, because the Web server 206 is unable to service  
16 requests for variable rate delivery of data. However, once the progressive  
17 download is complete, the playback module 220 enables the variable play speed  
18 controls 214 and continues to control the playback graph 218 to playback the  
19 media file 208 in accordance with variable play speed input from controls 214.

20 In an alternative implementation, the network layer 230 measures the  
21 average rate at which media file 202 is being progressively downloaded from Web  
22 server 106. The playback module 220 partially enables the variable play speed  
23 controls 214 to permit a user to request playback speeds that do not exceed the  
24 average download rate. For example, if the average download rate is 3.0x the real-  
25 time playback rate, the variable play speed controls 214 may allow the user to

1 request a playback speed in the range of 0.0x to 3.0x. In this example, playback  
2 speeds at rates greater than 3.0x would be disabled by the playback module 220.

3 When the playback module 220 queries the source layer 229 and determines  
4 that the media source is a streaming media server 104, it sends requests to the  
5 streaming media server 104 for variable rate data delivery that correspond with  
6 requests from the variable play speed controls 214 being input by a user. The  
7 playback module 220 generally maintains the variable play speed controls 214 in  
8 an enabled status unless there is a data delivery problem such as a bandwidth  
9 limitation. The playback module 220 communicates with a variable speed  
10 streaming module 212 on the streaming media server 104. The variable speed  
11 streaming module 212 is configured to respond to requests from the playback  
12 module 220 by accelerating or decelerating the delivery rate of data in a media file  
13 204. At any time before or during data delivery, if the server 104 bandwidth or  
14 network bandwidth become limited to the extent that accelerated delivery of media  
15 data is no longer possible, the playback module 220 may disable the variable play  
16 speed controls 214 on the media player 206. In this case, media playback would  
17 be maintained at a normal, real-time rate. In another variation, the playback  
18 module 220 can disable only relevant controls of the variable play speed controls  
19 214. For example, a user may be allowed to request playback at a rate that is  
20 slower than real-time, but might not be allowed to request playback at a rate that is  
21 faster than real-time. In yet another variation, the variable speed streaming module  
22 212 on the streaming media server 104 can disable relevant controls of the variable  
23 play speed controls 214 based on "policy" settings made, for example, by an  
24 administrator of the media server 104. For example, if the administrator of the  
25 media server 104 does not want the user to be able to fast-forward or seek through

1 a video advertisement, those buttons, including the variable speed controls, can be  
2 disabled in the GUI of the media player by communication with the playback  
3 module, even though the media stream is still delivered at an accelerated rate.

4 Alternatively, the playback module 220 can maintain the variable play speed  
5 controls 214 in an enabled state and gracefully degrade the quality of the playback.  
6 A graceful degradation of playback quality would result by the playback module  
7 220 first recognizing a data delivery limitation (e.g., limited network bandwidth,  
8 limited server 104 capacity) via network layer 230, and then requesting that the  
9 variable speed streaming module 212 in the streaming server 104 gracefully  
10 throttle back on the amount of data being delivered. The delivery rate of the data  
11 can be reduced to the normal or real-time bit-rate of the compressed media stream,  
12 but the variable speed streaming module 212 would, for example, deliver only  
13 video data and stop delivering audio data, or just deliver key frames (e.g., every 5<sup>th</sup>  
14 frame) of the video data. Thus, the variable play speed controls 214 on the media  
15 player 206 would remain enabled for use by a user, but the playback quality would  
16 be reduced. As soon as playback module 220 recognizes that the data delivery  
17 limitation (e.g., limited network bandwidth, limited server 104 capacity) has  
18 subsided, the playback module 220 can send a request to the streaming server 104  
19 to restore the playback quality (i.e., by increasing the data delivery rate) and enable  
20 the variable play speed controls 214 so that accelerated/variable speed playback  
21 can be resumed.

22 Figs. 5 and 6 help to illustrate how the playback module 220 manages the  
23 playback graph 218 to accelerate or decelerate the media data in a manner that  
24 maintains the ability of a user to comprehend the data. As mentioned above, there  
25 are various types of transform filters that can be alternately included in a playback

graph 218 to cause a particular desired effect in the playback of rendered data streams. Fig. 5 illustrates a playback graph 218 similar to that shown in Fig. 3, but which also includes an audio pitch adjustment filter 500 to process accelerated and decelerated audio data as it passes through the graph 218. One of the playback module's 220 tasks is to manage the playback graph 218 so that when it receives a request to vary the playback rate, it can ensure that the playback graph 218 has the appropriate assembly of filters to process the media data. Therefore, the playback module 220 controls the rate that data proceeds through the playback graph 218 and it also includes the audio pitch adjustment filter 500 for processing the accelerated or decelerated audio data. The audio pitch adjustment filter 500 is a time compression algorithm that makes it possible to have useful speed adjustments in audio playback.

Time compression is a technology that is generally well-known to those skilled in the art that permits changes in the playback rate of audio content without causing the pitch to change. Most systems today use linear time-compression algorithms, where audio/speech content is uniformly time compressed. In this class of algorithms, time-compression is applied consistently across the entire audio stream with a given speed-up rate, without regard to the audio information contained in the audio stream. Additional benefits can be achieved from non-linear time-compression techniques. Non-linear time compression is an improvement on linear compression where the content of the audio stream is analyzed and the compression rates may vary from one point in time to another. Typically, non-linear time compression involves an aggressive approach to compressing redundancies, such as pauses or elongated vowels. One such non-linear time-compression algorithm combines pause-removal with linear time

1 compression. It first detects pauses (i.e., silence intervals) in the audio/speech and  
2 then shortens or removes the pauses. Such a procedure can remove 10-25% from  
3 normal speech. It then performs linear time compression on the remaining speech.

4 Fig. 6, in conjunction with Fig. 4, help to illustrate how playing back (i.e.,  
5 rendering) all the content from a media stream without dropping data, such as  
6 dropping video frames, permits the playback of non-continuous, non-audio/video  
7 data that may be synchronized to particular locations within video data. Such data  
8 may include, for example, script commands for triggering events and other data  
9 streams such as captions and metadata.

10 In Fig. 6, the synchronized data stream represents non-audio/video text  
11 captions 600 that are synchronized for display within the video content at various  
12 locations. Such captions might be represented in Fig. 4 by the captions 418 shown  
13 in the video above the heads of the marathon runners. It is noted that such  
14 captions may be implemented in various ways, and that the illustrated form of the  
15 balloon text captions 418 shown in Fig. 4 is just one example of a possible  
16 implementation. Another likely example for implementing such text captions  
17 would be as simple text captions appearing in an area of the screen somewhere  
18 below the video display of Fig. 4. As shown in Fig. 6, when the media source  
19 (e.g., a streaming media server 104) can deliver data at an accelerated rate rather  
20 than having to degrade the quality of the playback by dropping data, all of the data  
21 in each of the video stream, the audio stream, and the synchronized data stream  
22 can be rendered or played back through playback graph 218. Thus, each point  
23 within the video stream where a synchronized text caption 600 occurs will result in  
24 the text caption 600 being displayed on the video display. The text captions 418 of  
25 Fig. 4 provide an example of captions 600 being displayed.

1 By contrast, if the media source (e.g., a streaming media server 104) cannot  
2 deliver data at an accelerated rate, but instead must degrade the quality of the  
3 playback by dropping video frames (i.e., delivering only key frames), then all of  
4 the data in the synchronized data stream may not be rendered or played back  
5 through playback graph 218. For example, if video data is dropped, resulting in  
6 only key video frames 602 being played back, then only the synchronized text  
7 captions 604 that occur with the key video frames 602 will be played back on the  
8 video display. The result may be that the text captions 418 of Fig. 4 would be left  
9 out of the playback. In an alternate implementation, the playback module 220  
10 controls playback graph 218 such that none of the synchronized data stream gets  
11 rendered during times when only key frames are being delivered for playback.

12 As mentioned above, media player 206 also includes a media player  
13 application programming interface (API) 224. In addition to the GUI module 222  
14 that maintains GUI 226 through which an end user has access to the variable play  
15 speed controls 214, the media player 206 also provides the media player API 224  
16 through which the variable play speed controls 214 are exposed to programmable  
17 control. The media player API 224 prescribes specific methods by which 3<sup>rd</sup> party  
18 software developers can access the variable play speed controls 214 of the media  
19 player 206 for use in custom application programs such as the 3<sup>rd</sup> party custom  
20 application program 232 illustrated in Fig. 2. The API 224 supports the various  
21 playback controls (214 and 216) for controlling media playback, including the play  
22 speed control, the content seek control, the fast forward control, the rewind  
23 control, the next frame and prior frame control, and the play/pause and stop  
24 controls. The API also supports capability flags to indicate when the media source  
25 can support accelerated data delivery rates.

## **Exemplary Methods**

Example methods for implementing variable play speed control of media streams will now be described with primary reference to the flow diagrams of Figs. 7 - 9. The methods apply generally to the exemplary embodiments discussed above with respect to Figs. 2 - 6. The elements of the described methods may be performed by any appropriate means including, for example, by hardware logic blocks on an ASIC or by the execution of processor-readable instructions defined on a processor-readable medium.

A "processor-readable medium," as used herein, can be any means that can contain, store, communicate, propagate, or transport instructions for use by or execution by a processor. A processor-readable medium can be, without limitation, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, device, or propagation medium. More specific examples of a processor-readable medium include, among others, an electrical connection (electronic) having one or more wires, a portable computer diskette (magnetic), a random access memory (RAM) (magnetic), a read-only memory (ROM) (magnetic), an erasable programmable-read-only memory (EPROM or Flash memory), an optical fiber (optical), a rewritable compact disc (CD-RW) (optical), and a portable compact disc read-only memory (CDROM) (optical).

Fig. 7 shows an exemplary method 700 for implementing variable play speed control of media streams on a media player 206. At block 702, a media player 206 renders media content at a real time playback rate. The real time playback rate is the normal rate at which the media content is intended to be consumed. The media content typically includes streams of audio and video data

1 but may also include other non-video/audio data such as script commands for  
2 triggering events, text captions, and metadata that is synchronized for rendering at  
3 particular times or locations within the video content.

4 At block 704 of method 700, the media player 206 receives a request to  
5 render the media content at an accelerated rate. The request is initiated either by  
6 an end user through a variable play speed control 214 of the media player 206, or it  
7 is initiated by a call to an application programming interface (API) 224 of the  
8 media player 206 from an application program 232.

9 At block 706, the media player 206 requests that the media content be  
10 delivered at the accelerated rate. The presumption in this case (i.e., method 700) is  
11 that the source of the media content is a streaming media server 104 that is capable  
12 of comprehending such requests for data delivery at variable rates, and, that the  
13 streaming media server 104 is capable of delivering data at the requested variable  
14 rates. At block 708, the media player 206 receives the media content at the  
15 requested accelerated rate. And at block 710, the media player 206 renders the  
16 media content at the accelerated rate.

17 Fig. 8 shows another exemplary method 800 for implementing variable play  
18 speed control of media streams on a media player 206. At block 802, a media  
19 player 206 receives a media stream from a source. The source is typically either a  
20 local media file 200 already residing on a client computer 102 on which the media  
21 player 206 is executing, a standard Web server 106 having media files 202  
22 available for downloading, or a streaming media server 104 having media files 204  
23 available for streaming delivery. At block 804, the media player 206 determines  
24 through a source layer 229, which of these devices is the source of the media  
25 content.

1       At block 806, if the source is not a local media file 200, the media player  
2 206 determines whether or not the source is capable of delivering the media stream  
3 at a variable rate. Note that if the source is a local media file 200, well-known  
4 delivery mechanisms within the client computer 102 are presumed to be able to  
5 deliver the media stream at an accelerated rate. The determination at block 806 is  
6 made by queries from a playback module 220 through a network layer 230 which  
7 can determine, for example, if a streaming media server 104 can respond to  
8 variable rate requests, if network bandwidth and other conditions will permit an  
9 accelerated delivery rate, and so on. In another variation where the source is a  
10 standard Web server 106, the network layer 230 in conjunction with the playback  
11 module 220 measures the average rate at which data arrives from the Web server  
12 106. If this rate is “accelerated”, compared to the normal playback rate, then the  
13 Web server 106 source is considered capable of delivering data at an accelerated  
14 rate. However, should the network conditions later deteriorate, the Web server 106  
15 source may be considered not capable of delivering the data at accelerated rates  
16 until the network conditions once again improve.

17       At block 808 of method 800, the media player 206 enables or disables  
18 variable play speed controls 214 of the media player 206 depending on the source  
19 and on whether the source is capable of delivering the media stream at the  
20 accelerated rate. Thus, for example, the variable play speed controls 214 may be  
21 disabled while a progressive file download occurs from a standard Web server 106  
22 if it is determined that the average data delivery rate is not “accelerated”,  
23 compared to the normal playback rate. However, once the download was  
24 completed, the controls would be enabled, because the file would then be a local  
25 media file which, as mentioned above, is capable of delivery at an accelerated rate

1 via known delivery mechanisms of the client computer 102. The variable play  
2 speed controls 214 can also be partially enabled/disabled depending on data  
3 delivery conditions. For example, if the average download rate from a Web server  
4 106 is 3.0x the real-time playback rate, the variable play speed controls 214 may  
5 be enabled to allow the user to request a playback speed in the range of 0.0x to  
6 3.0x. In this example, playback speeds at rates greater than 3.0x would be disabled  
7 by the playback module 220. Furthermore, where the data delivery rate does not  
8 permit an accelerated playback rate, the variable play speed controls 214 may still  
9 be enabled to allow the user to request a decelerated playback speed in the range  
10 of, for example, 0.0x to -2.0x.

11 Fig. 9 shows another exemplary method 900 for implementing variable play  
12 speed control of media streams on a media player 206. At block 902, a streaming  
13 media server 104 receives a request from a media player 206 executing on a client  
14 computer 102 to deliver media content at an accelerated rate. The accelerated rate  
15 is a rate beyond the normal, real time playback rate that the media content is  
16 intended to be consumed. At block 904, the streaming media server 104  
17 determines if it has the capacity to, and if the network link is able to, support the  
18 accelerated rate being requested. At block 906, the streaming media server 104  
19 delivers the media content to the client computer 102 at the accelerated rate if its  
20 capacity and the network link can support the accelerated rate. However, at block  
21 908, if either the streaming media server 104 capacity or the network link will not  
22 support the accelerated rate, then the streaming media server 104 can either drop  
23 data from the media content being requested (i.e., reduce the quality), or it can  
24 deliver the media content at the normal, real time playback rate of the media  
25 content. Dropping data from the media content may include dropping an audio

1 data stream and/or dropping video frames from video content such that only key  
2 video frames are delivered to the client computer 102.

3 While one or more methods have been disclosed by means of flow diagrams  
4 and text associated with the blocks of the flow diagrams, it is to be understood that  
5 the blocks do not necessarily have to be performed in the order in which they were  
6 presented, and that an alternative order may result in similar advantages.  
7 Furthermore, the methods are not exclusive and can be performed alone or in  
8 combination with one another.

### 9 10 **Exemplary Computer**

11 Fig. 10 illustrates an exemplary computing environment suitable for  
12 implementing a client computer 102, a streaming media server 104, and a standard  
13 Web server 106. Although one specific configuration is shown, client computer  
14 102, streaming media server 104, and standard Web server 106 may be  
15 implemented in other computing configurations.

16 The computing environment 1000 includes a general-purpose computing  
17 system in the form of a computer 1002. The components of computer 1002 can  
18 include, but are not limited to, one or more processors or processing units 1004, a  
19 system memory 1006, and a system bus 1008 that couples various system  
20 components including the processor 1004 to the system memory 1006.

21 The system bus 1008 represents one or more of any of several types of bus  
22 structures, including a memory bus or memory controller, a peripheral bus, an  
23 accelerated graphics port, and a processor or local bus using any of a variety of bus  
24 architectures. An example of a system bus 1008 would be a Peripheral Component  
25 Interconnects (PCI) bus, also known as a Mezzanine bus.

1 Computer 1002 typically includes a variety of computer readable media.  
2 Such media can be any available media that is accessible by computer 1002 and  
3 includes both volatile and non-volatile media, removable and non-removable  
4 media. The system memory 1006 includes computer readable media in the form of  
5 volatile memory, such as random access memory (RAM) 1010, and/or non-volatile  
6 memory, such as read only memory (ROM) 1012. A basic input/output system  
7 (BIOS) 1014, containing the basic routines that help to transfer information  
8 between elements within computer 1002, such as during start-up, is stored in ROM  
9 1012. RAM 1010 typically contains data and/or program modules that are  
10 immediately accessible to and/or presently operated on by the processing unit  
11 1004.

12 Computer 1002 can also include other removable/non-removable,  
13 volatile/non-volatile computer storage media. By way of example, Fig. 10  
14 illustrates a hard disk drive 1016 for reading from and writing to a non-removable,  
15 non-volatile magnetic media (not shown), a magnetic disk drive 1018 for reading  
16 from and writing to a removable, non-volatile magnetic disk 1020 (e.g., a "floppy  
17 disk"), and an optical disk drive 1022 for reading from and/or writing to a  
18 removable, non-volatile optical disk 1024 such as a CD-ROM, DVD-ROM, or  
19 other optical media. The hard disk drive 1016, magnetic disk drive 1018, and  
20 optical disk drive 1022 are each connected to the system bus 1008 by one or more  
21 data media interfaces 1026. Alternatively, the hard disk drive 1016, magnetic disk  
22 drive 1018, and optical disk drive 1022 can be connected to the system bus 1008  
23 by a SCSI interface (not shown).

24 The disk drives and their associated computer-readable media provide non-  
25 volatile storage of computer readable instructions, data structures, program

1 modules, and other data for computer 1002. Although the example illustrates a  
2 hard disk 1016, a removable magnetic disk 1020, and a removable optical disk  
3 1024, it is to be appreciated that other types of computer readable media which can  
4 store data that is accessible by a computer, such as magnetic cassettes or other  
5 magnetic storage devices, flash memory cards, CD-ROM, digital versatile disks  
6 (DVD) or other optical storage, random access memories (RAM), read only  
7 memories (ROM), electrically erasable programmable read-only memory  
8 (EEPROM), and the like, can also be utilized to implement the exemplary  
9 computing system and environment.

10 Any number of program modules can be stored on the hard disk 1016,  
11 magnetic disk 1020, optical disk 1024, ROM 1012, and/or RAM 1010, including  
12 by way of example, an operating system 1026, one or more application programs  
13 1028, other program modules 1030, and program data 1032. Each of such  
14 operating system 1026, one or more application programs 1028, other program  
15 modules 1030, and program data 1032 (or some combination thereof) may include  
16 an embodiment of a caching scheme for user network access information.

17 Computer 1002 can include a variety of computer/processor readable media  
18 identified as communication media. Communication media typically embodies  
19 computer readable instructions, data structures, program modules, or other data in  
20 a modulated data signal such as a carrier wave or other transport mechanism and  
21 includes any information delivery media. The term "modulated data signal" means  
22 a signal that has one or more of its characteristics set or changed in such a manner  
23 as to encode information in the signal. By way of example, and not limitation,  
24 communication media includes wired media such as a wired network or direct-  
25 wired connection, and wireless media such as acoustic, RF, infrared, and other

1 wireless media. Combinations of any of the above are also included within the  
2 scope of computer readable media.

3 A user can enter commands and information into computer system 1002 via  
4 input devices such as a keyboard 1034 and a pointing device 1036 (e.g., a  
5 "mouse"). Other input devices 1038 (not shown specifically) may include a  
6 microphone, joystick, game pad, satellite dish, serial port, scanner, and/or the like.  
7 These and other input devices are connected to the processing unit 1004 via  
8 input/output interfaces 1040 that are coupled to the system bus 1008, but may be  
9 connected by other interface and bus structures, such as a parallel port, game port,  
10 or a universal serial bus (USB).

11 A monitor 1042 or other type of display device can also be connected to the  
12 system bus 1008 via an interface, such as a video adapter 1044. In addition to the  
13 monitor 1042, other output peripheral devices can include components such as  
14 speakers (not shown) and a printer 1046 which can be connected to computer 1002  
15 via the input/output interfaces 1040.

16 Computer 1002 can operate in a networked environment using logical  
17 connections to one or more remote computers, such as a remote computing device  
18 1048. By way of example, the remote computing device 1048 can be a personal  
19 computer, portable computer, a server, a router, a network computer, a peer device  
20 or other common network node, and the like. The remote computing device 1048  
21 is illustrated as a portable computer that can include many or all of the elements  
22 and features described herein relative to computer system 1002.

23 Logical connections between computer 1002 and the remote computer 1048  
24 are depicted as a local area network (LAN) 1050 and a general wide area network  
25 (WAN) 1052. Such networking environments are commonplace in offices,

1 enterprise-wide computer networks, intranets, and the Internet. When  
2 implemented in a LAN networking environment, the computer 1002 is connected  
3 to a local network 1050 via a network interface or adapter 1054. When  
4 implemented in a WAN networking environment, the computer 1002 typically  
5 includes a modem 1056 or other means for establishing communications over the  
6 wide network 1052. The modem 1056, which can be internal or external to  
7 computer 1002, can be connected to the system bus 1008 via the input/output  
8 interfaces 1040 or other appropriate mechanisms. It is to be appreciated that the  
9 illustrated network connections are exemplary and that other means of establishing  
10 communication link(s) between the computers 1002 and 1048 can be employed.

11 In a networked environment, such as that illustrated with computing  
12 environment 1000, program modules depicted relative to the computer 1002, or  
13 portions thereof, may be stored in a remote memory storage device. By way of  
14 example, remote application programs 1058 reside on a memory device of remote  
15 computer 1048. For purposes of illustration, application programs and other  
16 executable program components, such as the operating system, are illustrated  
17 herein as discrete blocks, although it is recognized that such programs and  
18 components reside at various times in different storage components of the  
19 computer system 1002, and are executed by the data processor(s) of the computer.  
20

## 21 **Conclusion**

22 Although the invention has been described in language specific to structural  
23 features and/or methodological acts, it is to be understood that the invention  
24 defined in the appended claims is not necessarily limited to the specific features or  
25

1 acts described. Rather, the specific features and acts are disclosed as exemplary  
2 forms of implementing the claimed invention.  
3  
4  
5  
6  
7  
8  
9  
10  
11

12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25